

WiP Paper: Scattering Algorithm for Continuous Sensing in Battery-Free Networks

Johannes Göpfert
johannes.goepfert@tuhh.de
Hamburg University of Technology

Matteo Zella
matteo.zella@hsnr.de
Niederrhein University of Applied Sciences

Sayedsepehr Mosavat
sayedsepehr.mosavat@uni-due.de
University of Duisburg-Essen
sayedsepehr.mosavat@hsnr.de
Niederrhein University of Applied Sciences

Bernd-Christian Renner
christian.renner@tuhh.de
Hamburg University of Technology

Abstract

In the era of ubiquitous Internet of Things (IoT) devices and wearable technology, the sustainable operation of battery-free systems solely powered by energy harvesting presents both opportunities and challenges. We introduce *ScAlg*, a decentralized algorithm that leverages wake-up radios to scatter the active times of battery-free devices in a network, ensuring continuous operation and enhanced dependability even with tight energy constraints. Our approach leverages intermittent computing principles to reduce redundant activities and improve energy efficiency. By conducting simulations and hardware experiments, we demonstrate the algorithm's effectiveness. Although this is a work in progress, our findings show potential for strategic alignment of active times and intelligent task management in achieving sustainable, batteryless IoT systems capable of maintaining functionality in energy-variable environments.

CCS Concepts

• **Computer systems organization** → **Sensor networks**; • **Computing methodologies** → *Distributed algorithms*.

Keywords

Embedded systems, Energy harvesting, Battery-free sensor networks, Distributed sensing, Wake-up radios

1 Introduction

Rapid technological advancements have led to a growing concern about device energy consumption, particularly as they become more ubiquitous in our daily lives. Wearables, smartphones, and other IoT devices are now commonplace, and their proliferation has raised questions about sustainability because all of these aforementioned things of the internet are usually battery-powered. Battery replacement or recharging is a burden for the user and requires additional cyclic maintenance over the device's lifetime.

Intermittent computing (IC) is a paradigm that aims to address this issue by using energy harvesting to power devices. As the harvesting power is usually insufficient to power the device continuously, energy accumulates in a small capacitor. Once the capacitor is charged, the device can operate for a limited time before it runs out of energy.

However, this approach has its own set of challenges, as many common software and hardware development paradigms become void. For example, the assumption of a continuous and reliable power supply is no longer valid, and the system must be able to handle energy shortages gracefully. While [3, 7] present different methods for keeping a notion of time despite power outages. The methods presented in [19] improve the responsiveness of intermittent systems while maintaining forward progress. The feasibility of long-term monitoring without batteries has been proven by [2, 15].

Despite these advances, challenges remain. Due to the nature of intermittent systems and the fact that they are not always available, they are seen as unreliable for tasks with timely requirements like event detection. It can be challenging to detect an event with a single node, as the node might be inactive at the event's occurrence. Using multiple nodes increases the chances of detecting an event. However, they still might all be inactive at the same time, or a packet collision might occur if they try to indicate a detected event simultaneously. [12, 13] group a set of batteryless devices and try to distribute the active times so that the network can maintain a continuous operation despite intermittent devices. As energy is scarce, both works avoid communication and rely on heuristics and statistics to determine the active times of the co-located devices. While such attempts have shown significant improvements over the baseline without any shifting of active times, they still suffer because the devices are unaware of co-located devices. Rising technologies like wake-up radios (WuR) enable low-power one-directional communication, allowing energy-efficient coordination among devices [4]. Inspired by [10], which calls this procedure *scattering of wake-up times*, we propose a decentralized scattering algorithm that shifts the active times of batteryless devices in a network so that the network can maintain a continuous operation with an equidistant sampling rate. We exploit wake-up radios to scatter the devices' active times so that for a given task set that must be executed periodically, e.g., event detection, we can ensure bounds for the execution period depending on the devices' energy availability. This reduces redundant work, leading to lower channel occupancy. It also avoids packet collisions and lowers the chances of missed events, thus increasing the system's overall dependability.

This paper makes the following key contributions, which differentiate it from the state of the art:

- (1) We extend existing wake-up scattering approaches to batteryless sensor nodes by explicitly considering the energy intake and harvesting constraints.
- (2) We propose a solution applicable to batteryless sensors with both homogeneous and heterogeneous energy availability.
- (3) We develop a decentralized scheduling algorithm, *ScAlg*, which does not rely on statistical properties or heuristics, thereby enabling predictable and reliable operation.

Together, these contributions enable continuous and dependable sensor network operation by shifting nodes' active times with energy-aware scheduling using wake-up radios, thereby reducing packet collisions, lowering redundant communication, and increasing overall system dependability. We demonstrate the effectiveness of our approach through simulations and hardware experiments using the hardware presented in [15].

2 Related Work

The Coalesced Intermittent Sensor (CIS) concept, introduced by [13], combines multiple co-located intermittent sensor nodes into one virtual sensor. They claim that the devices' active times are distributed over time solely through inherent component variations. Based on this, the availability A_V of a CIS can be modeled as a function of the number of nodes N , a sensor's *on time* t_{on} , and its *power cycle time* t_p (i.e., the time between two consecutive power outages), given by

$$A_V(N) = 1 - \left(1 - \frac{t_{on}}{t_p}\right)^N. \quad (1)$$

Equation (1) shows that the availability of the sensor increases with a higher number of nodes or higher t_{on} of each node. One drawback of this approach is the assumption that all nodes experience the same amount of harvestable energy and that active times distribution arises only from hardware variations. This makes it difficult to predict the sensor availability dependably.

In contrast, our work adopts the CIS concept but exploits energy-efficient communication via wake-up radios to deliberately align the active times of nodes in a predictable manner, also supporting heterogeneous energy availability among nodes.

Islam et al. [12] propose two methods for scattering the active times of co-located devices. Their first method, Prime-Co-Prime Duty Cycle selection, targets scenarios where nodes have static but heterogeneous energy availability. Duty cycles are pre-assigned based on prime numbers, requiring prior knowledge of relative energy availability and synchronized clocks. While this leads to better utilization than uniform duty cycles, redundant active times can still occur, and the approach cannot adapt to runtime energy fluctuations. Our algorithm adapts dynamically to changing energy availability and strictly avoids redundant active periods.

Their second method targets scenarios with dynamically varying energy availability. To address this, they formulate the problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), a framework for decision-making in multi-agent systems under partial observability. The authors propose a set of heuristics—including Randomized Binary Search (RBS) and Suboptimal Reinforcement Learning (SRL)—to adapt duty cycles online according to the available energy. In contrast, our algorithm does

not rely on heuristics or learning methods, yet it effectively adapts to changing energy conditions in a deterministic and predictable manner.

Other works [8, 10] leverage communication to scatter node wake-up times based on the fireflies-inspired synchronization model by Mirollo and Strogatz [14]. Although these algorithms ensure no redundant active times, they result in uniform duty cycles across all nodes. This uniformity is energy efficient only if all nodes share the same amount of harvestable energy, which is often not the case. Our approach improves upon this by allowing energy-aware, heterogeneous duty cycling that increases overall energy efficiency.

In summary, unlike prior approaches such as [10], which do not consider energy intake and harvesting constraints, our method explicitly incorporates these factors. Additionally, our solution supports batteryless sensors with both homogeneous and heterogeneous energy availability, addressing limitations found in [10], [12] and [13]. Finally, by developing the decentralized scheduling algorithm *ScAlg*, which does not rely on statistical properties or heuristics, we enable more reliable and predictable operations compared to related works [12] and [13].

3 Algorithm

ScAlg is a distributed algorithm for energy-aligned cyclic task execution in a network of sensor nodes. The algorithm assumes that all nodes within the network can communicate with each other and periodically perform the same set of tasks. Furthermore, the nodes require a mechanism for time synchronization and are able to determine their own sampling period as stated in Eq. (4). We use wake-up radio messages for (relative) time synchronization and communication. The sampling period is determined by a voltage supervisor connected to the energy buffer and the MCU's internal RTC. Of course, other methods for time synchronization and sampling period determination are possible. For the purposes of this work, *ScAlg* is designed for single-hop, static networks where all nodes are within communication range of each other. Multi-hop or mobile scenarios are not considered here, as they require more complex coordination mechanisms which are beyond the scope of the present work.

3.1 Energy Model for Cyclic Task Execution

This section presents the mathematical model for energy-aligned cyclic task execution for a single sensor node. We consider a sensor node in soft-IC configuration, which means that the node does not turn off when energy is depleted but instead enters a low-power state [18]. For an energy-neutral cyclic task execution, it holds that

$$E_{\text{harv}} = E_{\text{tasks}} + E_{\text{sleep}}, \quad (2)$$

where E_{harv} is the energy harvested during a cycle, E_{tasks} is the energy consumed by the tasks, which in our case comprises the MCU and its peripherals, and E_{sleep} is the energy consumed during sleep. Assuming a constant sleep and harvesting power P_{sleep} and P_{harv} , at least for the duration of one period T , it holds that

$$P_{\text{harv}} \cdot T = P_{\text{sleep}} \cdot t_{\text{sleep}} + \sum_{i=1}^n P_i \cdot t_i. \quad (3)$$

Table 1: Comparison of related scattering algorithms

Feature	Majid et al. [13]	Islam et al. [12]	Firefly inspired [8, 10]	ScAlg (this work)
Energy assumption	Homogeneous	Heterogeneous, dynamic	Not accounted for	Heterogeneous, dynamic
Duty cycle	Implicit	Prime / Dec-POMDP	Uniform	Adaptive, energy-aware
Energy adaptation	✗	✓ (heur./learning)	✗	✓ (deterministic)
Requires communication	✗	✗	✓	✓
Strictly avoids redundancy	✗	✗	✓	✓

Where P_i is the power consumption of task i and t_i is the task's execution time, both are known beforehand. We can further simplify this relation by using $T = t_{\text{sleep}} + \sum_{i=1}^n t_i$. For easier handling we substitute $\sum_{i=1}^n P_i \cdot t_i = P_{\text{tasks}} \cdot t_{\text{tasks}}$ with $P_{\text{tasks}} = (\sum_{i=1}^n P_i \cdot t_i) / t_{\text{tasks}}$ and $t_{\text{tasks}} = \sum_{i=1}^n t_i$. With this and Eq. (3) we can calculate the cycle period as

$$T = \frac{P_{\text{tasks}} - P_{\text{sleep}}}{P_{\text{harv}} - P_{\text{sleep}}} \cdot t_{\text{tasks}}. \quad (4)$$

This shows that the cycle period T can be determined by the tasks' power consumption P_{tasks} and execution times t_{tasks} , the sleep power consumption P_{sleep} and the harvesting power P_{harv} . Note that P_{harv} must be larger than P_{sleep} for obtaining a positive cycle period. The size of the energy storage does not affect the cycle period; it is only used to buffer the harvested energy.

3.2 Execution Model for Distributed Sensing

For modeling distributed sensing, we start with the case where the network comprises N nodes that harvest the same amount of energy. Then, according to Eq. (4), all nodes will have the same cycle period T . If all nodes are aware of each other, they can shift their execution times by a fixed amount of time $\Delta t = T/N$ relative to each other. This means that the nodes will not execute the task at the same time but rather in a round-robin fashion. A distributed sample period can then be calculated with

$$T_d = \frac{T}{N}. \quad (5)$$

Now, we remove the constraint that nodes harvest the same amount of energy, leading to different cycle periods T_i for each node. Nodes with higher energy availability will have shorter cycle periods than nodes with lower energy availability. Two nodes with different cycle periods could wake up simultaneously after a certain amount of cycles [12]. To avoid this and energy depletion for a node, all nodes could adjust to the same sample period T_s , where T_s is the period of the slowest node in the network. Afterward, the nodes shift T_s/N relative to each other, as described previously.

For these two scenarios, wake-up times could be scattered using the aforementioned algorithms based on the Mirolo-Strogatz (Firefly) model. However, when nodes have different energy availability, this has one essential drawback. The distributed sample period T_d can get larger for N nodes than for $N - 1$ nodes. More specifically, $T_d(N) > T_d(N - 1)$, leading to N nodes doing less work than $N - 1$ nodes. Using Eq. (5), we can show that this is the case if the following condition holds:

$$T_s(N) > N \cdot T_d(N - 1). \quad (6)$$

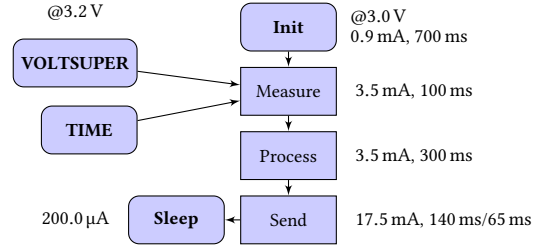


Figure 1: Measure, process and send cycle used for the evaluation of ScAlg. Sub-tasks are noted with their respective execution times and current consumption. Task execution is either triggered by a voltage supervisor or on a temporal basis depending on the nodes energy availability.

For example, a network with three nodes has a sample period $T_s(4 - 1) = 1$ s leading to a distributed sample period of $T_d(4 - 1) = 1/3$ s. If a fourth node with sample period $T_s(4) = 3$ s joins the network, $T_d(4) = 3/4$ s $>$ $T_d(4 - 1) = 1/3$ s. To tackle this problem, we need to consider that nodes might be able to sample more than once within a period T_s . In the following, we will call this behavior *intersampling*, and with this, we introduce the intersampling rate R_i . R_i indicates the number of times that node i can execute a given set of tasks within the period T_s without energy depletion. In terms of cycle periods according to Eq. (4) we can calculate the intersample rate as:

$$R_i = \frac{T_s}{T_i}. \quad (7)$$

For ScAlg, we will exploit this relation to calculate the intersampling rate for each node within the network and scatter each node's task execution accordingly within the slowest node's period T_s . We furthermore extend Eq. (7) with a communication task that is not executed R_i times but only once per node per cycle.

3.3 Implementation

ScAlg relies on the distribution of a node's sampling period, stated in Eq. (4), to all other nodes in the network. The nodes calculate their intersampling rate R_i and shift their sampling time(s) relative to each other. As an example, we consider the scenario where a set of nodes is supposed to execute a measure-process-send cycle periodically, as shown in Fig. 1. All nodes send their processed data to a sink node, where the received data is aggregated. The time and current consumption values are examples but are based on real measurements. For further calculations, we assume that the voltage is constant during task execution, leading to a constant

power consumption for each task. The send task has two different execution times depending on how much data is sent. The long execution time is used when the node broadcasts its ID, followed by the sampling period plus the data for the sink. The short execution time is used when the node only sends its data to the sink. In this case, other nodes do not receive any data. We differentiate between these two cases to be more energy efficient when the intersampling mechanism gets active.

Init phase: In the initial phase of *ScAlg*, the nodes are not aware of each other nor their sampling period. Therefore, the first step is determining the node's sampling period. In our case, this is done by starting the measure-process-send cycle via an interrupt triggered by a voltage supervisor (3.2 V). After the node executes the measure-process-send cycle, it sleeps until the next interrupt is triggered. The node measures the time between two consecutive interrupts to determine its own sampling period. During this phase, the node receives wake-up messages from other nodes and processes them but does not yet react to them. In the initial phase, we neglect this extra energy consumption. This error vanishes when the network stabilizes during the operational phase because the slowest node keeps measuring its sample period.

Operational phase: Once the node has determined its sample period, it starts broadcasting it and enters the operational phase where the wake-up time scattering starts. Over time, a node becomes aware of all the other nodes in the network and their sampling periods. Hence, every node can maintain a list of all known nodes in the network and their sampling periods. In descending order, we sort this list by the node's possible sampling period. If nodes have the same sampling period, we also sort the list by the node's ID to avoid an ambiguous order. Thus, we can assume that $T_{i+1} \leq T_i$, $i \in 1, \dots, N$. This will lead to all nodes having exactly the same list, which is important for determining a unique relative shift for each node. With this list, the nodes calculate the intersampling rates R_i based on the slowest sampling period $T_s = T_1$. We extend the R_i calculation from Eq. (7) with the processing task when a broadcasted sampling rate from another node is received. This means that within the period T_s the node needs to have enough energy intake to execute the measure-process-send cycle R_i times and the reception task $N - 1$ times, once for receiving from each of the other $N - 1$ nodes. Using Eq. (4) and Eq. (7) we can then infer that the intersampling rate R_i for node i can be calculated with:

$$R_i = \left\lfloor \frac{1}{a-b} \left(\frac{T_s}{T_i} \cdot (a + (N-1) \cdot b) - N \cdot b \right) \right\rfloor \quad (8)$$

with :

$$a = (P_{\text{tasks}} - P_{\text{sleep}}) \cdot t_{\text{tasks}}$$

$$b = (P_{\text{recv}} - P_{\text{sleep}}) \cdot t_{\text{recv}}$$

Note that by flooring the value we consider the maximum integer value for R_i that the node can operate in without energy depletion. With all R_i , we can calculate the distributed sampling period with

$$T_d = \frac{T_s}{\sum_{i=1}^N R_i} \quad (9)$$

Finally, we determine how to shift the nodes. Therefore, we take the node with the slowest sampling period T_s as the reference node. It will continue executing the measure-process-send cycle triggered

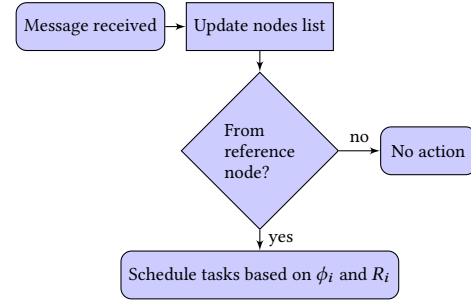


Figure 2: Update procedure of *ScAlg* on reception of a wake-up message.

by the voltage supervisor interrupt and update its sampling period. With this approach, the network adapts quickly to changes in the energy availability of the reference node. The other nodes start sampling on a temporal base and shift their active times relative to it according to the calculated R_i . These nodes disable the voltage supervisor interrupt but keep broadcasting their last measured sampling period. This way, new nodes joining the network can adapt, and failing nodes can be detected by the missing messages. For synchronization, we use the reception time of the message the reference node sends. On reception of the wake-up message, all nodes calculate their relative shift with:

$$\phi_i = \left(\sum_{k=1}^{i-1} R_k \right) \cdot T_d - t_{\text{tasks}} \quad (10)$$

With Eq. (10), the nodes start sampling in the order they are listed in the nodes list, considering the intersample rate of the preceding nodes. We subtract the tasks' execution times, considering that the message is sent after the reference node's task execution. The update and shift procedure is also shown in Fig. 2. An example for the resulting shifts ϕ_i and intersampling rates R_i is shown in Fig. 3.

4 Evaluation

To evaluate the performance of *ScAlg* and demonstrate its feasibility, we conduct experiments using both hardware and simulation.

We use the hardware presented in [15], which provides features that simplify the implementation of *ScAlg*. It uses a solar cell as an energy harvesting source, connected to a maximum power point (MPPT) controller. With this architecture, P_{harv} does not depend on the voltage of the energy storage (here a capacitor with 47 mF), which in turn depends on the (previous) power consumption. This leads to a steadier P_{harv} , which is advantageous for *ScAlg*.

Furthermore, it features a wake-up receiver and a transceiver. The wake-up receiver supports three addressing modes: unicast, multicast, and broadcast, enabling selective wake-up. In addition, it provides extra data reception of up to five bytes, facilitating broadcasts. The latter comes in handy for the implementation of *ScAlg* as it allows the broadcast of the sampling period without turning on the primary transceiver. Furthermore, it enables the nodes to synchronize on incoming messages because the WuR signals a wake-up ID match via a GPIO pin that triggers an interrupt on the MCU. According to the datasheet [9], its current consumption

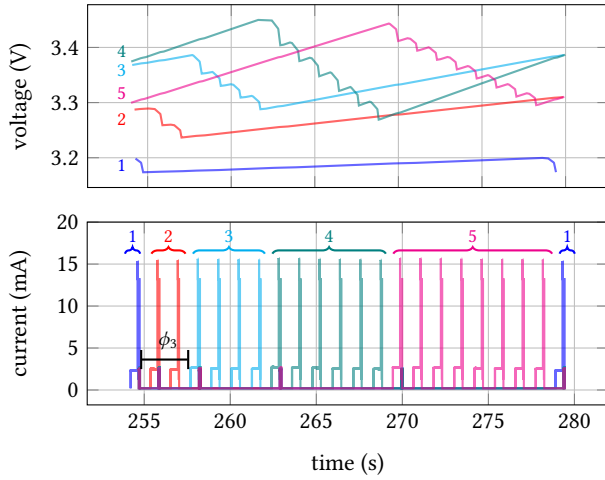


Figure 3: Simulated capacitor voltages (above) and current consumptions (below) of five sensor nodes running *ScAlg* with differing charging powers ranging from 1 mW (Node 1) to 3 mW (Node 5). The nodes’ wake-up times are scattered to distribute the workload.

is $\leq 3.5 \mu\text{A}$, making data reception cheaper, from an energy perspective, compared to traditional transceivers. For comparison, the current consumption of the Transceiver CC1101 is around 15 mA in RX mode [17].

The simulation results are obtained with a simulator based on NS-3 [1]. A detailed sensor node model is given in [11]. The mathematical model for the energy household is presented in [6]. For communication, we use NS-3’s spectrum framework [5].

4.1 Simulation Results

Figure 3 shows the simulated capacitor voltages and current consumptions of five sensor nodes running *ScAlg* after scattering the nodes’ wake-up times. In this scenario, each node has a different charging power, starting from 1 mW for node 1, increasing by 0.5 mW with each subsequent node up to node 5 with 3 mW. From the current consumption we can infer when a particular node executes a task. Lower current values around 2.5 mA correspond to measurement and processing tasks, while peaks indicate data transmission. Spikes following the transmission phases represent data reception and processing from other nodes. Node 1 is the slowest node with a sampling period $T_s \approx 25$ s resulting in a distributed sampling period of $T_d \approx 1.19$ s. An increase in charging power from 1 mW to 1.5 mW for node 2 already enables an intersampling rate $R_2 = \lfloor 2.7 \rfloor = 2$, reflected in two current peaks. Node 5, with 3 mW charging power, achieves $R_5 = \lfloor 8.2 \rfloor = 8$. The voltage plot shows that node 1 triggers task execution when the capacitor voltage reaches 3.2 V. Other nodes operate on time-triggered schedules with voltages above this threshold, consistent with their inability to exploit the full charging power immediately. Consequently, voltage levels rise slowly over time if the intersampling rate is not an exact integer.

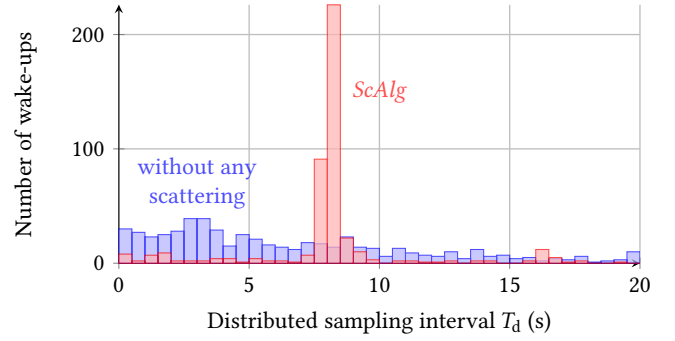


Figure 4: Wake-up distribution for a 2 h experiment involving five sensor nodes, each powered with 1 mW. The histogram shows the number of occurrences of the distributed sampling interval T_d with a bin width of 0.5 s. With the scattering algorithm (*ScAlg*, red) enabled, measurement events are more uniformly distributed, reducing simultaneous wake-ups and collisions. Without scattering (blue), measurement events tend to occur in close temporal proximity, resulting in overlapping wake-ups and increased network contention.

4.2 Experimental Results

We used a testbed consisting of five sensor nodes for our hardware experiment. Each node received a constant charging power of approximately 1 mW from a solar cell emulator as described in [16].

Figure 4 shows the statistical distribution of the distributed sampling period T_d during approximately 2 h of operation, comparing scenarios with and without *ScAlg*. Although all nodes were supplied with the same charging power, the sample periods varied between roughly 20 s and 40 s due to hardware variances among individual nodes. When *ScAlg* is enabled, nodes successfully scattered their wake-up times, resulting in a more uniform distribution of measurement events with an average distributed sampling interval T_d of about 8.12 s and a standard deviation of 2.86 s. In contrast, without scattering, the nodes’ wake-up times spread significantly, leading to an average $T_d = 6.40$ s and a standard deviation of 4.96 s. While nodes running *ScAlg* completed 440 samples during the test period, nodes without scattering sampled more frequently (560 times), which appears advantageous at first glance. However, the more frequent sampling without scattering leads to increased packet collisions, redundant transmissions, and higher channel occupancy. *ScAlg* avoids these issues by enforcing equidistant sampling times, trading off aggressive sampling for improved efficiency and reliability of network operation.

4.3 Analysis and Insights

The benefits of *ScAlg* are evident in both simulation and hardware experiments. The results demonstrate how *ScAlg* effectively distributes the active times of sensor nodes, allowing them to operate efficiently even with varying energy harvesting capabilities.

As *ScAlg* relies on communication between nodes, it is essential to consider the energy overhead associated with this communication. The energy overhead is primarily caused by sending a wake-up

message once per cycle, processing the $N - 1$ received wake-up messages and the wake-up receiver's energy consumption while listening for wake-up messages. Therefore, the energy overhead can be calculated with,

$$E_{\text{overhead}} = P_{\text{WuR}} \cdot T_s + (N - 1) \cdot P_{\text{process}} \cdot t_{\text{process}} + P_{\text{send}} \cdot t_{\text{send}} \quad (11)$$

For further comparison, we take the values from our hardware and our use case from Section 3, i.e., $P_{\text{WuR}} = 10.5 \mu\text{W}$, $P_{\text{process}} = 4.5 \text{ mW}$, $t_{\text{process}} = 20 \text{ ms}$, and $P_{\text{send}} = 52.5 \text{ mW}$, $t_{\text{send}} = 75 \text{ ms}$. With these values, considering a sampling period of $T_s = 25 \text{ s}$, we can denote the energy overhead of *ScAlg* for a network of five nodes with 4.56 mJ per sampling period. With a charging rate of 1 mW we have 25 mJ available per sampling period, which means that the energy overhead of *ScAlg* is 18.2% of the available energy. This amount could be further reduced by reducing the wake-up message broadcasts to every k -th sampling period or broadcasting only on considerable changes in energy availability, which would reduce the number of wake-up messages sent and received.

In contrast, without *ScAlg* 18% of the sample periods are lower than 1.9 s . Leading to redundant data, higher channel occupancy and a higher probability of packet collisions. Additionally, around 32% of the wake-ups without scattering happen with a sampling period higher than the average distributed sampling period of *ScAlg* (8.12 s), which could lead to missed events or data.

The obtained average distributed sampling period of *ScAlg* is 26.8% longer than the average sampling period without scattering, which reflects the energy overhead of *ScAlg*. Therefore, *ScAlg* reduces the standard deviation of the sampling period by 42.3% and the IQR (interquartile range) reduces from 6.6 s to 0.4 s (93.2%). These reduction values indicate that *ScAlg* effectively reduces the variance in sampling periods, leading to a more stable and predictable sampling behavior in the network.

5 Conclusion

We present our ongoing work *ScAlg*, a distributed scheduling algorithm that leverages wake-up radios to scatter node active times in battery-free networks. Our simulations and hardware experiments show that *ScAlg* significantly reduces sampling variability (42.3% lower standard deviation, IQR reduced by 93.2%) while increasing the average sampling interval by about 26.8% . These reduction values indicate that *ScAlg* effectively reduces the variance in sampling periods, leading to a more stable and predictable network behavior. The energy overhead of *ScAlg* is moderate, increasing sampling energy by roughly 18.2% in our setup, which remains acceptable given the benefits.

Our current work focuses on refining the algorithm and exploring additional optimization techniques, such as adaptation to rapidly changing energy harvesting conditions or the distribution of the inter-sample times. The latter is not optimal as it requires a larger energy storage than necessary. Extending *ScAlg* to multi-hop and larger-scale networks is another promising avenue to enhance its applicability. We plan to make *ScAlg*'s source code publicly available upon completion of our work.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the "DI2T" project (project number 492151098).

References

- [1] [n. d.]. *ns-3 Network Simulator*. <https://www.nsnam.org> last accessed 2024-01-24.
- [2] Mikhail Afanasov, Naveed Anwar Bhatti, Dennis Campagna, Giacomo Caslini, Fabio Massimo Centonze, Koustabh Dolui, Andrea Maioli, Erica Barone, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, and Luca Mottola. 2020. Battery-less zero-maintenance embedded sensing at the mithraeum of circus maximus. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (Virtual Event, Japan) (*Sensys '20*). ACM, 368–381. doi:10.1145/3384419.3430722
- [3] Arwa Alsubhi, Nicole Tobias, Simeon Babatunde, and Jacob Sorber. 2020. Can Crystal Oscillators Keep Time Without Power?. In *Proceedings of the 8th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems* (Virtual Event, Japan) (*ENSys '20*). ACM, 84–85. doi:10.1145/3417308.3430278
- [4] Simeon Babatunde, Arwa Alsubhi, Josiah Hester, and Jacob Sorber. 2024. Green-tooth: Robust and Energy Efficient Wireless Networking for Batteryless Devices. *ACM Trans. Sen. Netw.* 20, 3, Article 66 (April 2024), 31 pages. doi:10.1145/3649221
- [5] Nicola Baldo and Marco Miozzo. 2009. Spectrum-aware channel and PHY layer modeling for ns3. In *VALUETOOLS '09*.
- [6] Hannah Brunner, Jasper de Winkel, Carlo Alberto Boano, Przemysław Pawelczak, and Kay Römer. 2024. Simba: A Unified Framework to Explore and Facilitate the Design of Battery-Free Systems. In *23rd ACM/IEEE International Conference on Information Processing in Sensor Networks* (Hong Kong, China) (*IPSN '24*). 138–150. doi:10.1109/IPSNS61024.2024.00016
- [7] Jasper de Winkel, Carlo Delle Donne, Kasim Sinan Yildirim, Przemysław Pawelczak, and Josiah Hester. 2020. Reliable Timekeeping for Intermittent Computing. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) (*ASPLOS '20*). ACM, 53–67. doi:10.1145/3373376.3378464
- [8] Julius Degesys, Ian Rose, Ankit Patel, and Radhika Nagpal. 2007. DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks* (Cambridge, Massachusetts, USA) (*IPSN '07*). ACM, 11–20. doi:10.1145/1236360.1236363
- [9] Fraunhofer IIS. 2023. *RFicient® Ultra-low Power Wake-Up Receiver FH101RF*. Fraunhofer IIS. Rev. 1.3, preliminary.
- [10] Alessandro Giusti, Amy L Murphy, and Gian Pietro Picco. 2007. Decentralized scattering of wake-up times in wireless sensor networks. In *Wireless Sensor Networks: 4th European Conference, EWSN 2007, Delft, The Netherlands, January 29-31, 2007. Proceedings 4*. Springer, 245–260.
- [11] Johannes Göpfert and Bernd-Christian Renner. 2023. High-level Simulation of the Timely Behavior of Intermittent Systems. In *Proceedings of the 11th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems* (Istanbul, Türkiye) (*ENSys '23*). ACM, 1–7. doi:10.1145/3628353.3628539
- [12] Bashima Islam, Yubo Luo, and Shahriar Nirjon. 2023. Amalgamated Intermittent Computing Systems. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation* (San Antonio, TX, USA) (*IoTDI '23*). ACM, 184–196. doi:10.1145/3576842.3582388
- [13] Amjad Yousef Majid, Patrick Schilder, and Koen Langendoen. 2020. Continuous Sensing on Intermittent Power. In *19th ACM/IEEE International Conference on Information Processing in Sensor Networks* (Sydney, Australia) (*IPSN '20*). IEEE, 181–192. doi:10.1109/IPSNS48710.2020.00-36
- [14] Renato E. Mirolo and Steven H. Strogatz. 1990. Synchronization of Pulse-Coupled Biological Oscillators. *SIAM J. Appl. Math.* 50, 6 (1990), 1645–1662. arXiv:https://doi.org/10.1137/0150098 doi:10.1137/0150098
- [15] Sayedsepehr Mosavat, Matteo Zella, Marcus Handte, Alexander Julian Golkowski, and Pedro José Marrón. 2023. Experience: ARISTOTLE: Wake-up Receiver-Based, STar Topology BaTteryLEss Sensor Network. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks* (San Antonio, TX, USA) (*IPSN '23*). ACM, 177–190. doi:10.1145/3583120.3586961
- [16] Sayedsepehr Mosavat, Matteo Zella, and Pedro José Marrón. 2021. Demo: SOCRATES: SOLar Cells Recorded And Emulated EaSily. In *Proceedings of the 2021 International Conference on Embedded Wireless Systems and Networks* (Delft, The Netherlands) (*EWSN '21*). Junction Publishing, USA, 183–184.
- [17] Texas Instruments. 2013. *Low-Power Sub-1 GHz RF Transceiver*. Texas Instruments. SWRS061I.
- [18] Mathew L. Wymore and Henry Duwe. 2023. A Tale of Two Intermittencies. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems* (Boston, Massachusetts) (*Sensys '22*). ACM, 928–930. doi:10.1145/3560905.3568105
- [19] Kasim Sinan Yildirim, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemysław Pawelczak, and Josiah Hester. 2018. InK: Reactive Kernel for Tiny Batteryless Sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems* (Shenzhen, China) (*Sensys '18*). ACM, 41–53. doi:10.1145/3274783.3274837